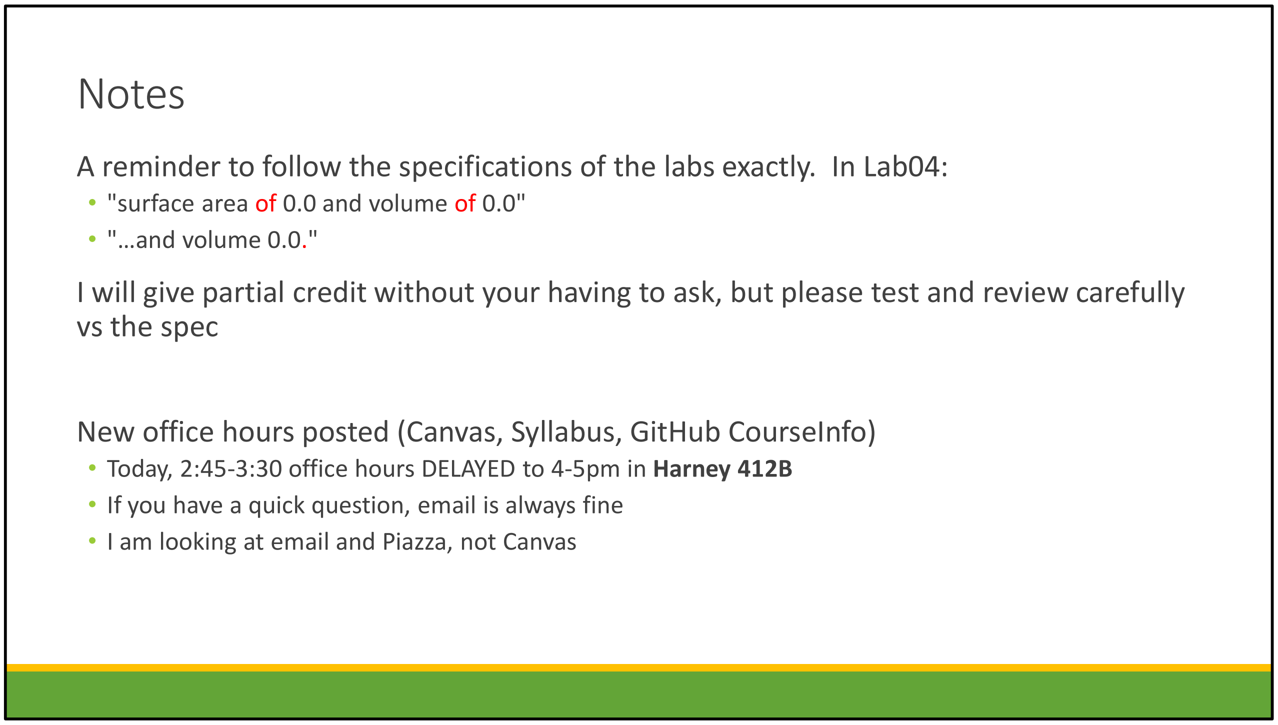
CS112 –
Java
Programming

Fall 2022

Java Objects and References



Notes

A reminder to follow the specifications of the labs exactly. In Lab04:

- "surface area of 0.0 and volume of 0.0"
- "...and volume 0.0."

I will give partial credit without your having to ask, but please test and review carefully vs the spec

New office hours posted (Canvas, Syllabus, GitHub CourseInfo)

- Today, 2:45-3:30 office hours DELAYED to 4-5pm in **Harney 412B**
- If you have a quick question, email is always fine
- I am looking at email and Piazza, not Canvas

The "of" and the "." should not be there.

My idea for Project01: all write programs to play BlackJack together, via Internet connections. All must follow command protocol EXACTLY.

Last time…

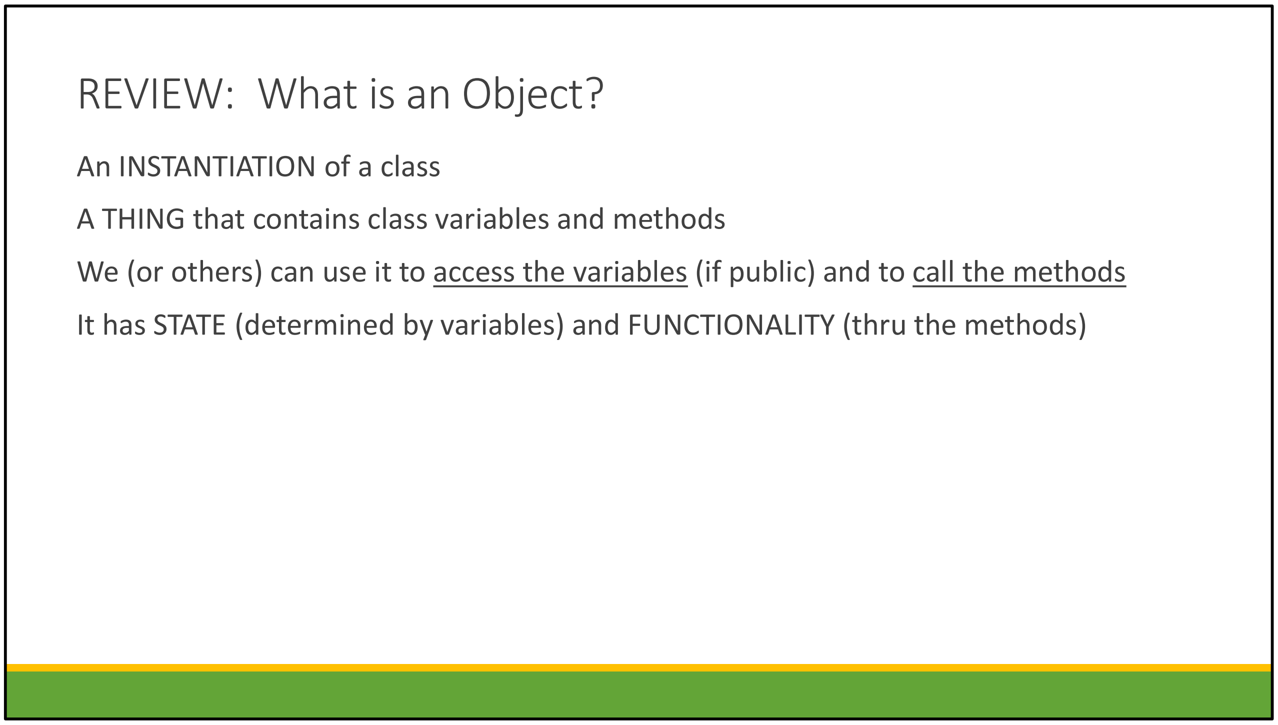
We discussed Java classes and objects

We discussed class member variables and methods

- Method return types, method arguments
- "get()" methods and "set()" methods

We experimented with class String

We saw how to access command-line arguments to Java programs



REVIEW: What is an Object?

An INSTANTIATION of a class

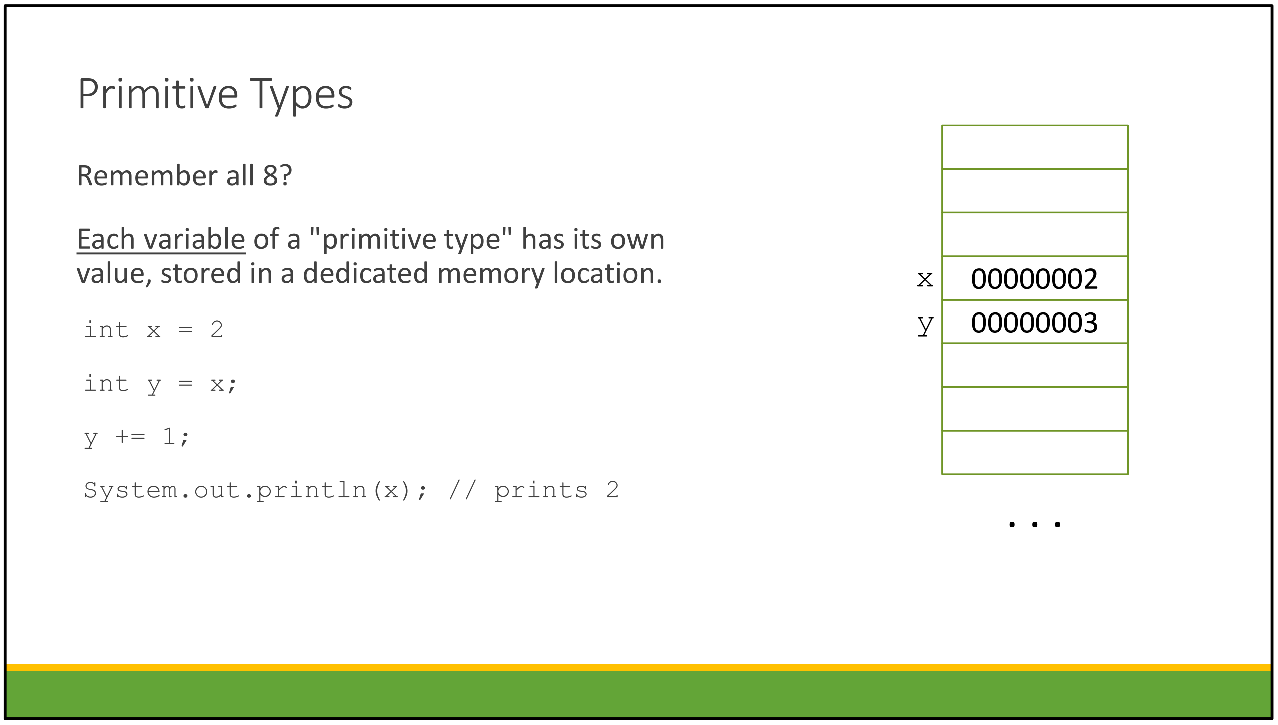
A THING that contains class variables and methods

We (or others) can use it to access the variables (if public) and to call the methods

It has STATE (determined by variables) and FUNCTIONALITY (thru the methods)

LOOK AT Border.java

Primitive Types vs Object Types

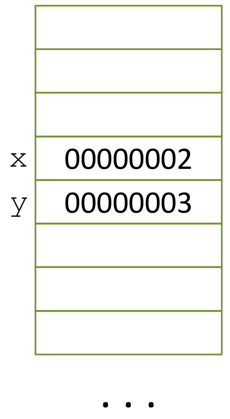


Primitive Types

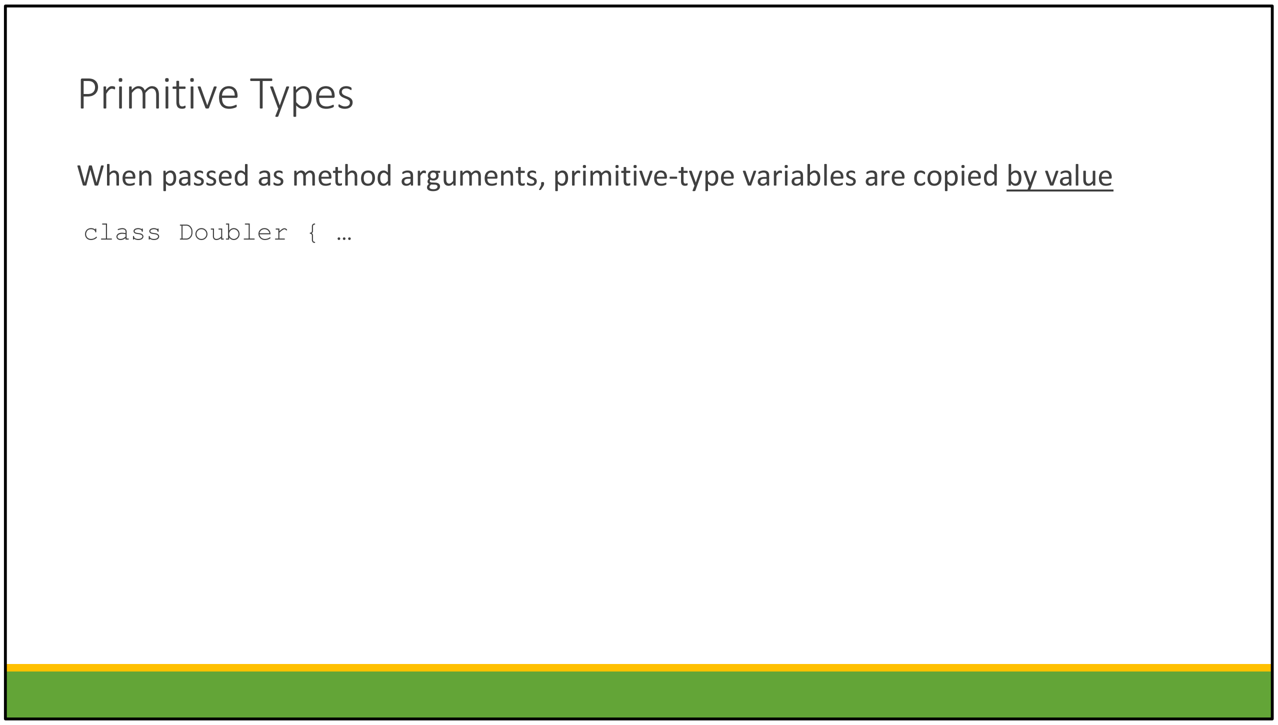
Remember all 8?

Each variable of a "primitive type" has its own value, stored in a dedicated memory location.

```
int x = 2

int y = x;

y += 1;

System.out.println(x); // prints 2
```



Byte short int long float double char boolean



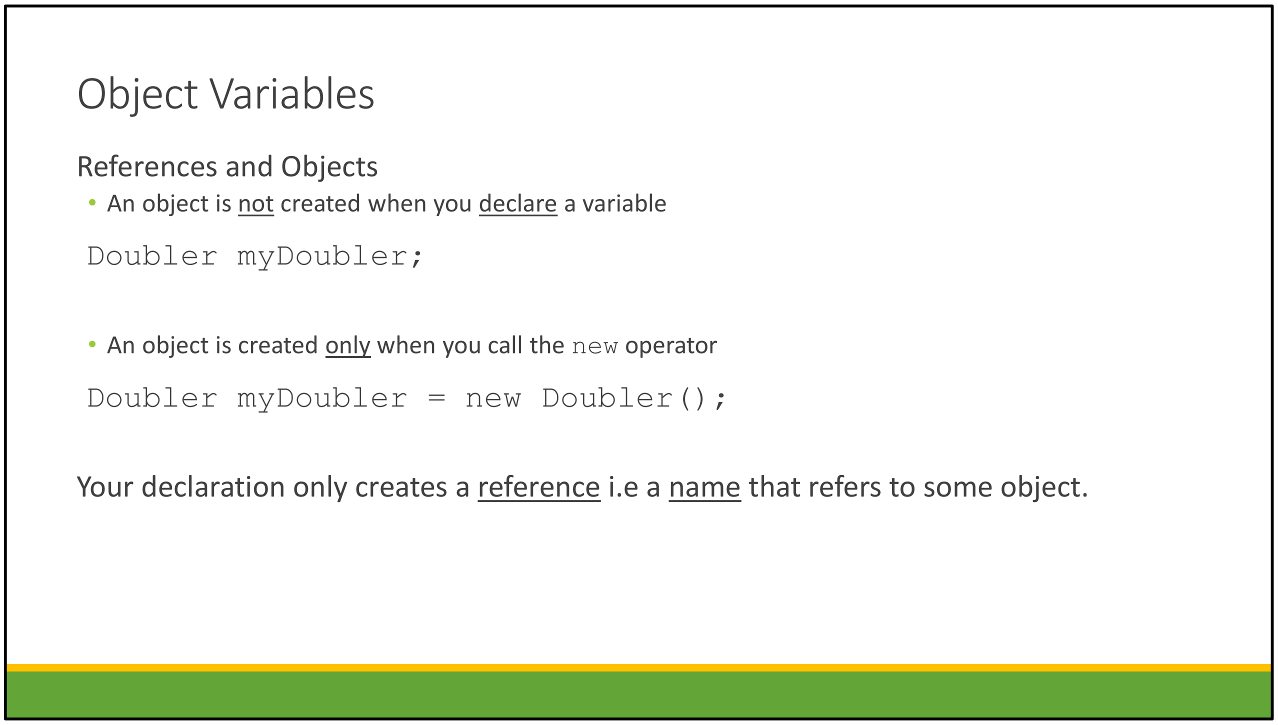
Primitive Types

When passed as method arguments, primitive-type variables are copied by value

```
class Doubler { …
```

Do in ECLIPSE:
class Doubler { double value; void set(double x) { x *= 2; value = x; } double get() { return value; }}

int y = 4; myDoubler = new Doubler(); myDoubler(x); print(y)



Object Variables

References and Objects

- An object is not created when you declare a variable

```
Doubler myDoubler;
```

- An object is created only when you call the `new` operator

```
Doubler myDoubler = new Doubler();
```

Your declaration only creates a reference i.e a name that refers to some object.

Object Variables

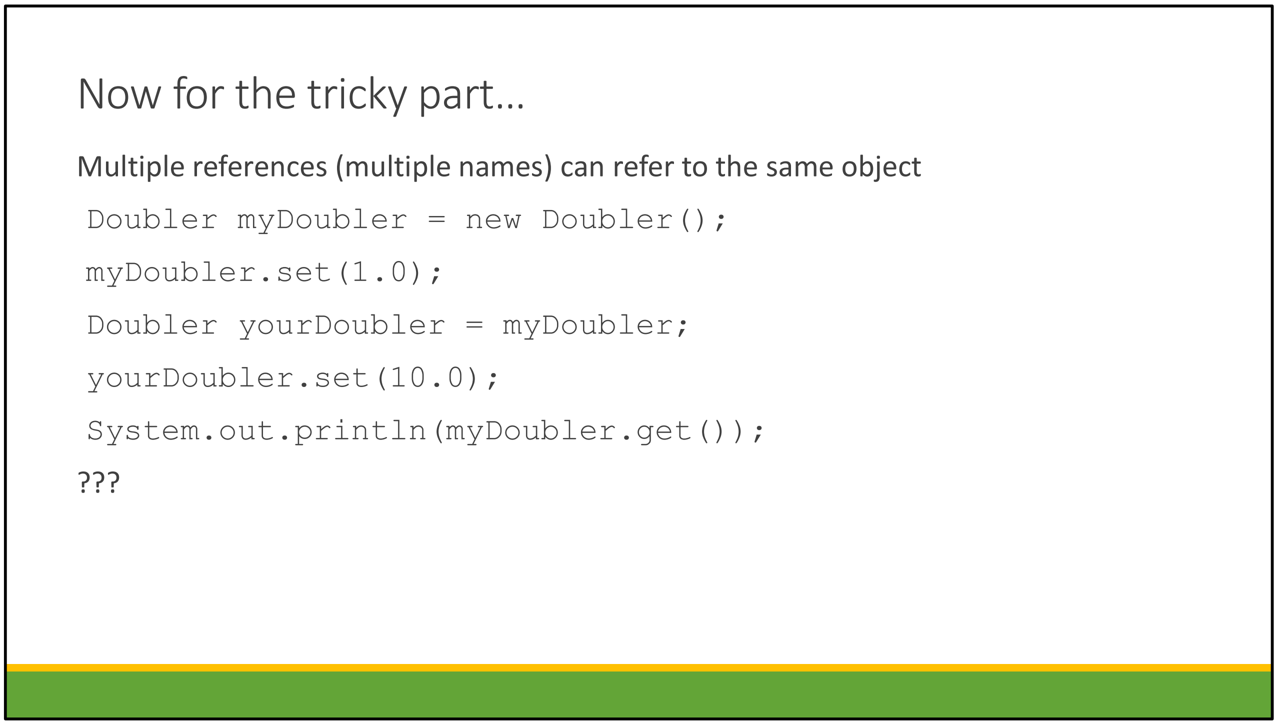
Ok then, what object is referred to by

```
Doubler myDoubler;
```

???

The reference is to a special Java value called `null`.

- You can check if a reference is null. `if (myDoubler == null) { …`
- You can assign `null` to a reference.
- But of course you cannot call any class methods on `null`, cannot assign values to `null`, etc.



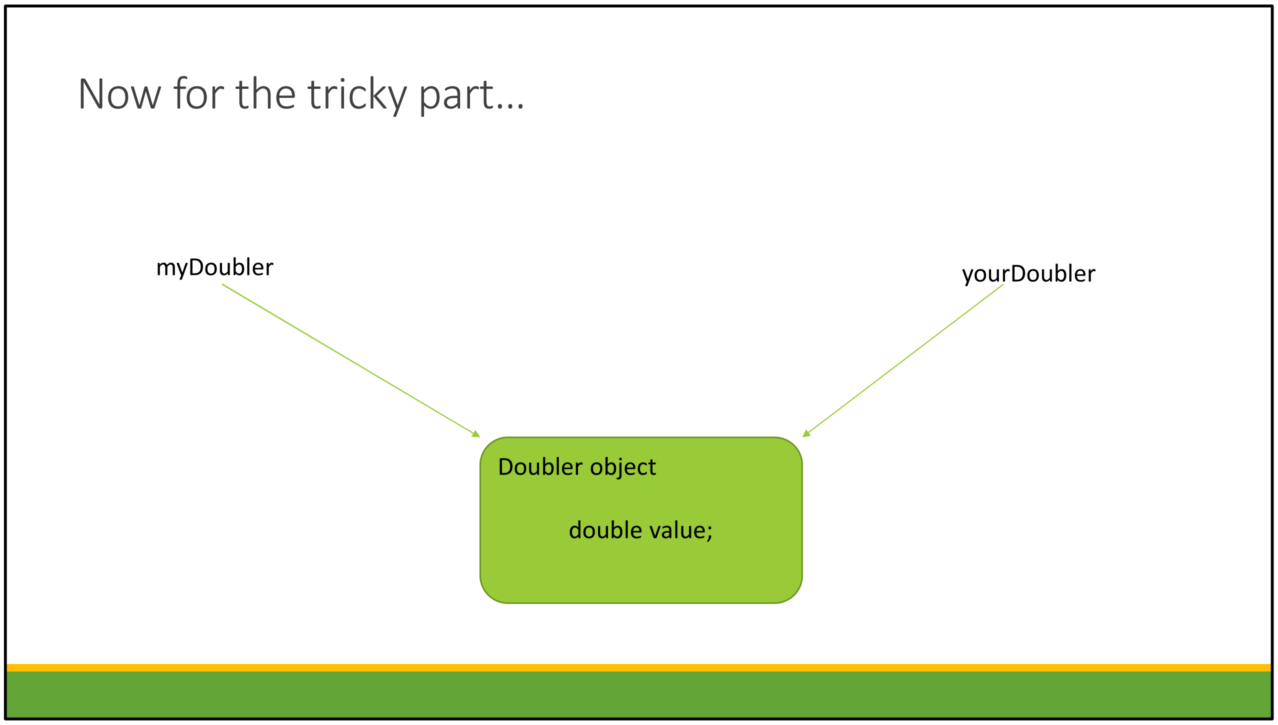
Now for the tricky part…

Multiple references (multiple names) can refer to the same object

```
Doubler myDoubler = new Doubler();
myDoubler.set(1.0);
Doubler yourDoubler = myDoubler;
yourDoubler.set(10.0);
System.out.println(myDoubler.get());
```

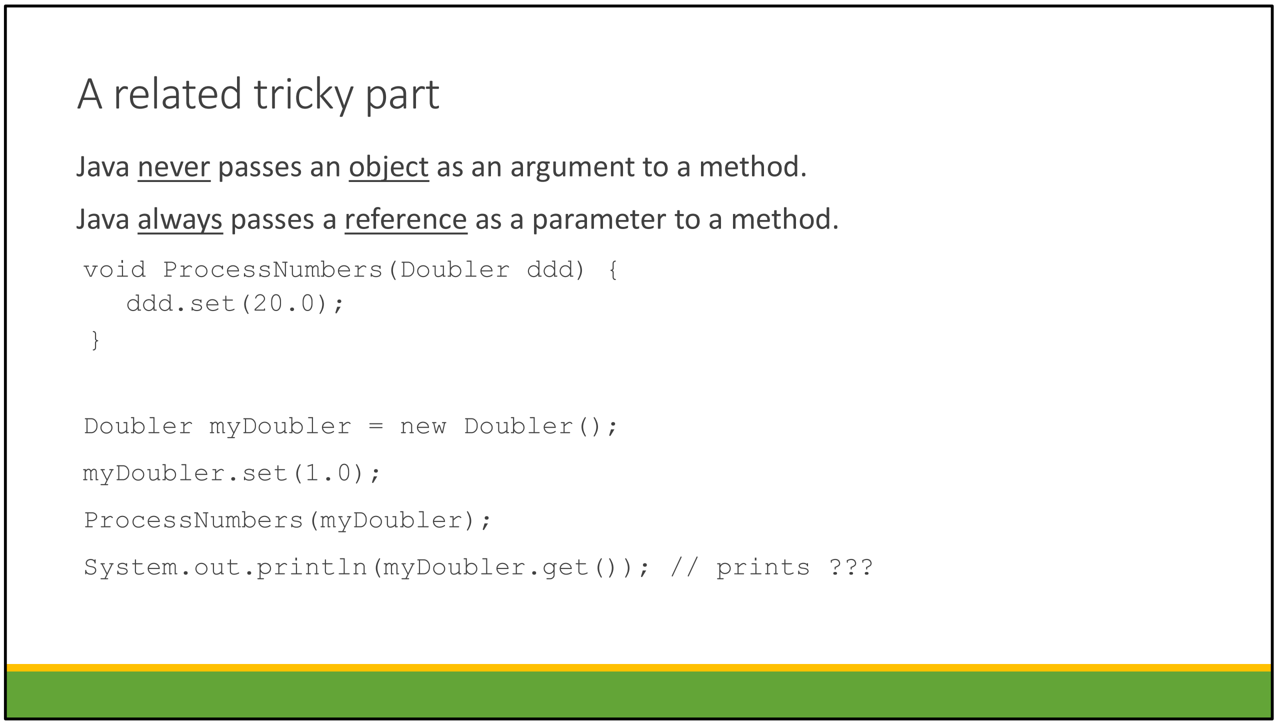
???

Prints 20



Now for the tricky part…

THIS DOES NOT HAPPEN WITH PRIMITIVE DATA TYPE VARIABLES: THEY JUST HOLD A VALUE.



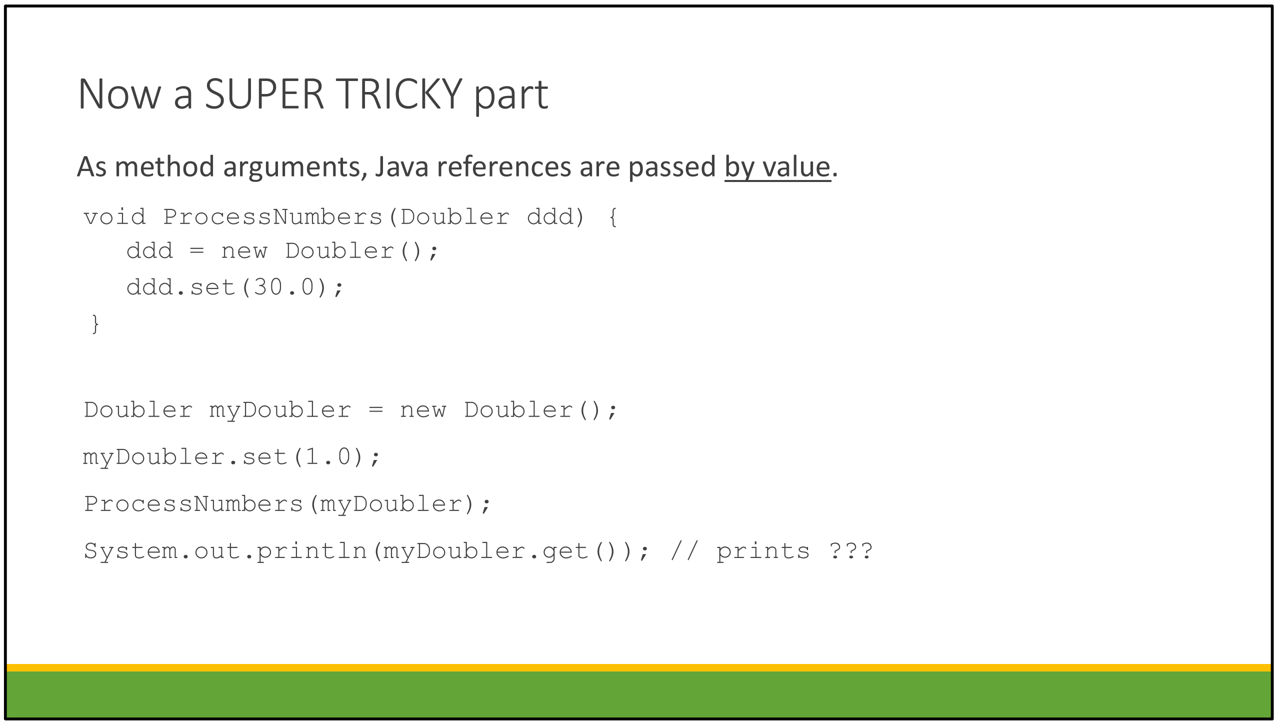
A related tricky part

Java never passes an object as an argument to a method.

Java always passes a reference as a parameter to a method.

```
void ProcessNumbers(Doubler ddd) {
   ddd.set(20.0);
 }


Doubler myDoubler = new Doubler();

myDoubler.set(1.0);

ProcessNumbers(myDoubler);

System.out.println(myDoubler.get()); // prints ???
```

Prints 40.0. DRAW IT OUT!



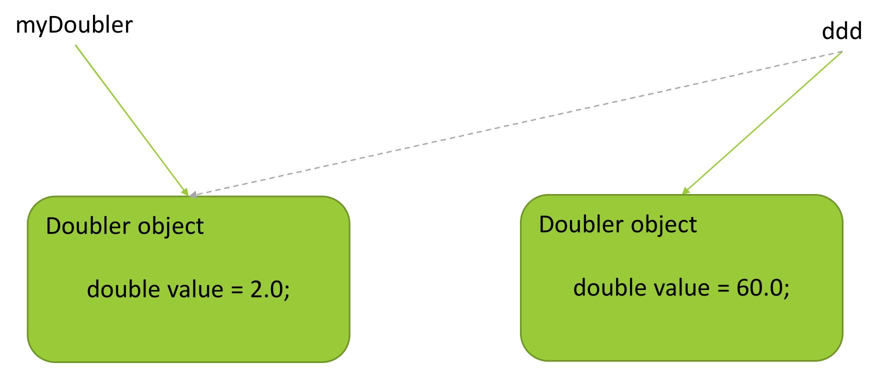
Now a SUPER TRICKY part

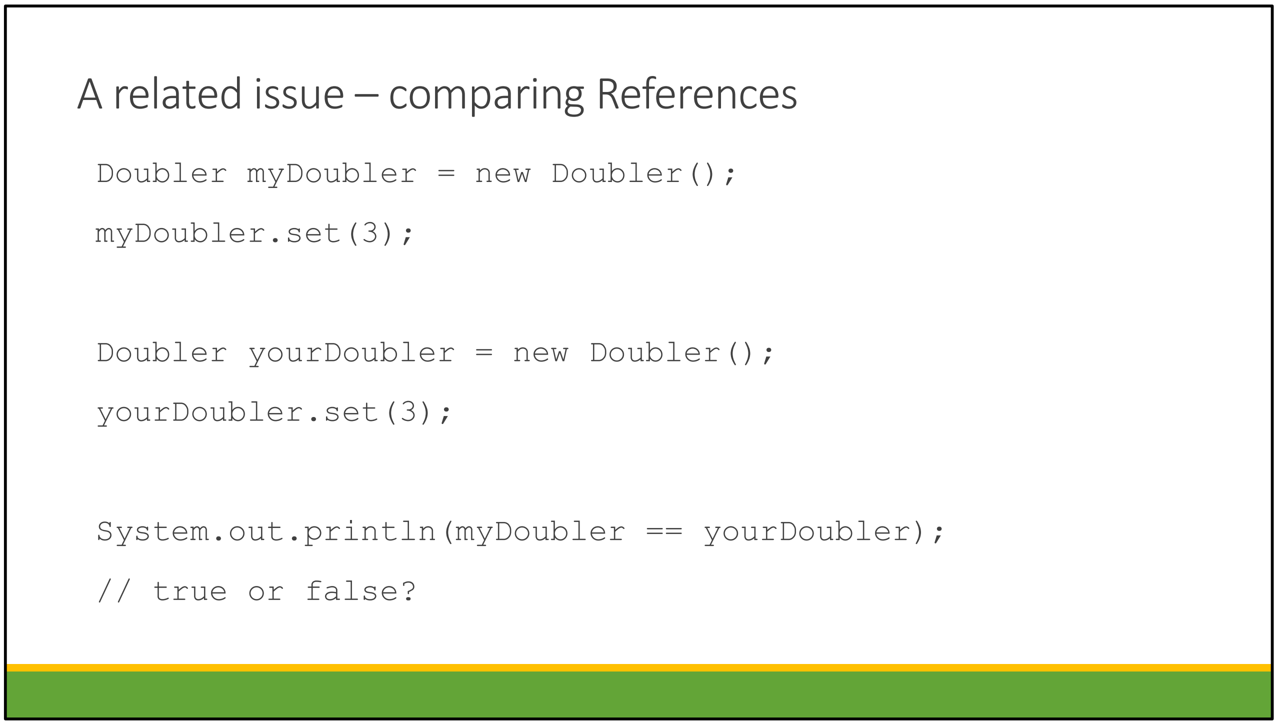
As method arguments, Java references are passed by value.

```
void ProcessNumbers(Doubler ddd) {
   ddd = new Doubler();
   ddd.set(30.0);
}


Doubler myDoubler = new Doubler();
myDoubler.set(1.0);
ProcessNumbers(myDoubler);
System.out.println(myDoubler.get()); // prints ???
```

Prints 2.0. DRAW IT OUT!
THIS IS TRICKY!

Now a SUPER TRICKY part

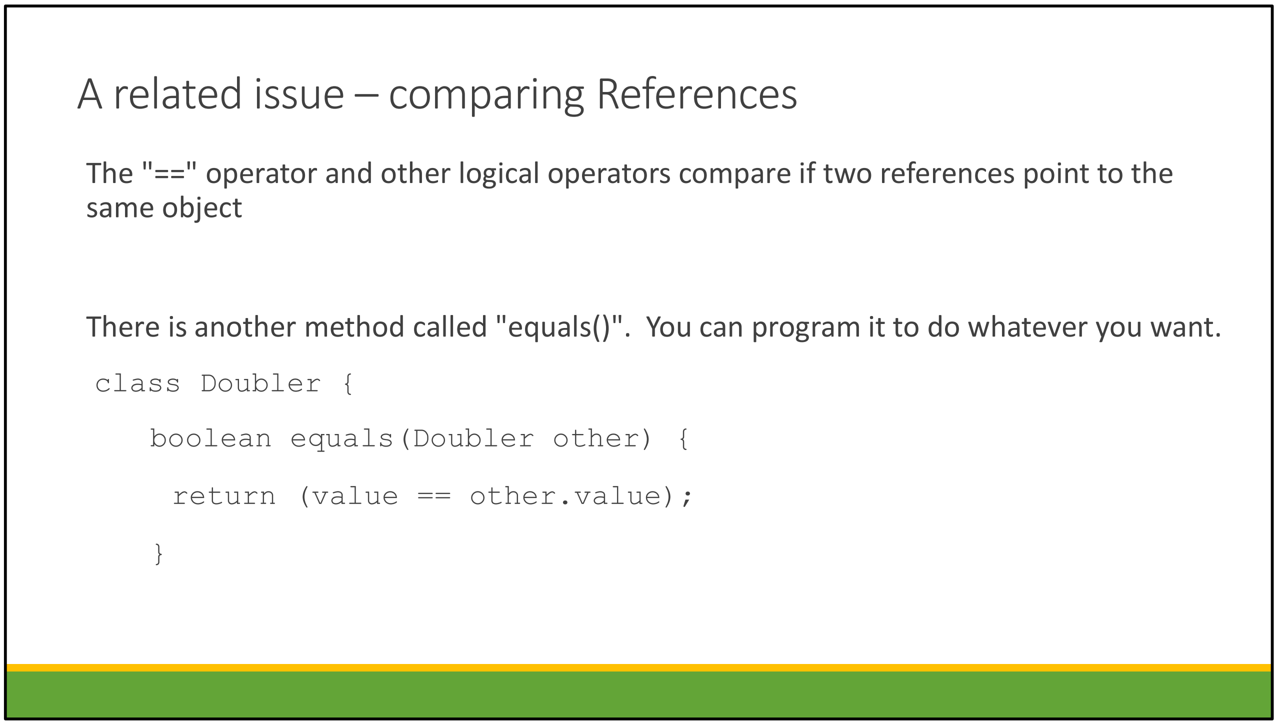




A related issue – comparing References

```
Doubler myDoubler = new Doubler();
myDoubler.set(3);

Doubler yourDoubler = new Doubler();
yourDoubler.set(3);

System.out.println(myDoubler == yourDoubler);
// true or false?
```

false



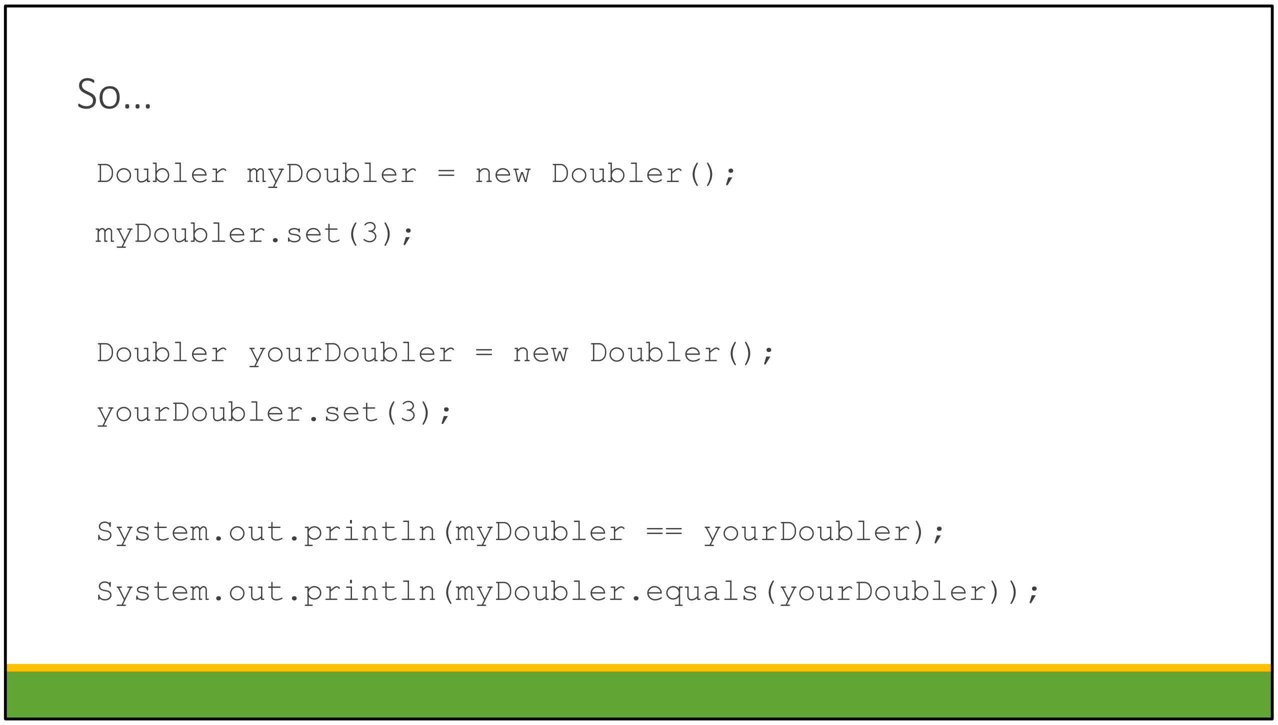
A related issue – comparing References

The "==" operator and other logical operators compare if two references point to the same object

There is another method called "equals()". You can program it to do whatever you want.

```
class Doubler {
    boolean equals(Doubler other) {
      return (value == other.value);
    }
```

return (other.value == 26) is legal, but really confusing.



So…

```
Doubler myDoubler = new Doubler();
myDoubler.set(3);

Doubler yourDoubler = new Doubler();
yourDoubler.set(3);

System.out.println(myDoubler == yourDoubler);
System.out.println(myDoubler.equals(yourDoubler));
```

False, True

Quiz

Lab06

And we can look at Lab05 also